High Performance Tag Singulation for Memory-less RFID Systems

Keyvan Kashkouli Nejad
Tohoku University, Sendai, Japan, JSPS fellow.
Email: keyvan@ecei.tohoku.ac.jp.

Xiaohong Jiang
Future University Hakodate, Japan.
Email: jiang@fun.ac.jp

Michitaka Kameyama
Tohoku University, Sendai, Japan.
Email: kameyama@ecei.tohoku.ac.jp.

***Abstract*—The simple and cheap memory-less RFID tag systems have found many diverse applications. The available singulation algorithms for such systems adopt either single-slot or multi-slot frames with a fixed frame size, independent of tag population. A single-slot scheme has good performance in terms of the number of overall slots N, but it can cause a very large number of requests R. A multi-slot scheme, on the other hand, achieves good performance of R, but it may introduce an unacceptable large N. In this paper, we propose an adaptive frame size singulation scheme for memory-less RFID systems. Our basic idea is to adaptively choose a suitable frame size based on tag population information collected during the singulation, such that the number of unnecessary collision and idle slots can be reduced. Analytical models are also derived for the performance analysis of the new scheme. Surprisingly, our theoretical and simulation studies indicated that through adopting such adaptive frame size, the new singulation scheme can achieve a performance of N as good as the single-slot scheme, and at same time, guarantees a performance of R similar to the available multi-slot schemes.**

***Index Terms*—RFID, memory-less tags, tag singulation algorithm.**

I. Introduction

Radio frequency identification (RFID) is an automatic identification system consisting of readers and tags, where each tag has unique identification number (ID) so the reader can recognize objects through consecutive communications with tags attached to them. The typical applications of RFID technology include inventory management, real-time monitoring, etc. Recently, the RFID technology has also found new applications in location discovery [1], [2], popular items identification [3], traffic tracking [4], etc.

A common problem of any RFID systems is tag singulation for the purpose of tags counting or identifying, where counting is the process of finding exact population cardinality, while identifying is to retrieve the unique information about each object embedded in the tag. In the singulation process a reader sends a request (probe) towards the tags, and tags reply to the request in one or multiple slots based on a specified singulation algorithm. Collision happens if multiple tags reply to a request in the same slot. The reader needs to send additional requests to arbitrate collisions until at most one tag replies in a slot.

The RFID tags can be generally classified as memory-less ones (like the passive tags) and memory-based ones (like the semi-passive and active tags). The key difference between these two classes is that a memory-based tag has its own memory and power source, typically in the form of a battery. With such memory, the tags can store some feedback information in the singulation process, like the information of reader's inquiries and/or the response of the tag itself, which can significantly help the tag singulation algorithm in collision arbitration. The available singulation algorithms for memory-based tags can be found in [5]–[8]. It is notable, however, the memory requirement in RFID tags significantly increases equipment cost, and more importantly, the small battery in such a tag can not be recharged, which largely limits the lifetime of the tag. This paper focus on the tag singulation in memory-less RFID system, because it is much cheaper, has a very long lifetime (no battery lifetime problem), and also has diverse and promising applications.

In a memory-less RFID system, the tag singulation can not rely on tags' memory of previous feedback information for collision arbitration. Instead, the tags' responses are solely determined by the reader's current inquiry, and the reader can only prevent a tag from replying by sending a prefix that should match the ID of the tag. Thus, all tags in a memory-less RFID system will continue to transmit in every frame as long as they match the probe, so the total time needed to recognize all tags can be very long [9]. Therefore, the fast and low energy consumption singulation of massive tags in such system is now a challenging issue. Some singulation schemes have been proposed for memory-less RFID tag systems [10]–[13] (Please refer to the related work in Section. II-B). In general, these schemes can singulate all tags without relying on the feedback messages from reader, thus omitting the memory requirement for storing such information. Nonetheless, these schemes always adopt a fixed number of reply slot(s) in each frame (called $frame\ size$ hereafter) regardless of the tag population, so they usually result in either a large number of overall slots N or a large number of requests R.

In this paper, we first review in Section II related work on tag singulation for memory-less RFID systems, then we propose in Section III an adaptive frame size (AFS) singulation scheme for memory-less RFID systems. The scheme can adaptively choose a suitable frame size based on tag population information collected during the singulation process. Analytical models are further developed in Section IV for the performance of the new scheme. We demonstrate through theoretical and simulation results in Section V that

This work is partially supported by JSPS Grant (B) 21300018.

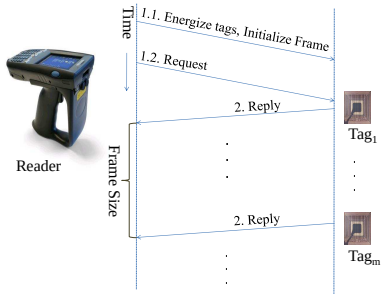


Fig. 1. Illustration of a RFID singulation frame

the proposed scheme can guarantee a desirable performance in terms both N and R.

II. Memory-less RFID System and Related Work

A. System Model

In the RFID system a reader is responsible for singulating a set of tags within its communication range. Each tag has its identification number (ID) in the form of a binary sequence, and the reader usually performs tag singulation through frame by frame communication with tags as shown in Fig. 1. From the figure we can see that each singulation frame actually involves several key phases, such as tag energizing and frame initialization, reader querying, and tags replying. These phases can be virtually categorized in two main sections, querying section and reply section. In the querying section, the reader first needs to send out signals to energize and synchronize all tags, it then sends out a querying request to tags which typically includes the prefix information in form of a bit string. The reply section of the frame consists of one/multiple time slot(s) for the tags' replies. Upon receiving the request, a tag whose ID match with the prefix will select a slot for sending its information. After receiving the replies, the reader needs to determine the state of each allocated slot (collision, readable or idle). Since a passive tag cannot detect collision, the reader tries to arbitrate collisions by sending further queries towards the tags. This process continues until all tags in the reading range of the reader are recognized.

Since in each frame the reader needs to send out one request (querying request) to query tags, which is usually energy consuming, so the overall energy consumption will be controlled by the number of singulation frames (and thus the number of requests R). As the querying section for specified RFID system usually adopts a fixed format and thus unchangeable, so we can only change the number of slots in the reply section (i.e., the frame size) to control the overall number of slots N and also R, which together determine the overall singulation delay.

B. Related Work on Memory-less Tag Singulation

Some interesting schemes have been proposed for tag singulation in memory-less RFID systems, see for example, [10]–[13]. In these schemes, by representing each collision, readable (singleton) and empty slot as a collision, readable (singleton) and empty node in a singulation tree, the singulation problem can then be transformed into a readable nodes searching problem in the tree.

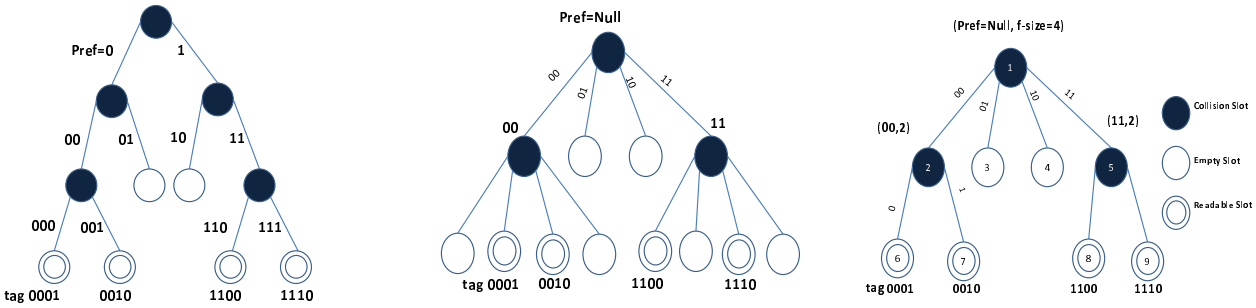
Some of available singulation schemes allocate only one slot in each frame for tags to reply, like [10], [14], and we call such schemes as single slot (SS) schemes hereafter. These SS schemes usually incur a small N, because they only visit least number of empty (idle) nodes in the singulation tree. It is notable, however, these algorithms suffer from a large number of query requests R, and that's why some recent schemes apply multiple but fixed number of reply slots in each frame [15], [16] (such schemes are called multiple slot (MS) schemes hereafter). The MS schemes can usually reduce collisions and the number of the requests R. However, adopting a large and unsuitable number of reply slots in each frame may likely introduce larger number of idle slots. Here, we introduce in more details the representative SS scheme Query Tree (QT) [10] and MS scheme Hybrid Query Tree (HQT) [15].

1) Query Tree Scheme, QT: The operation of QT scheme is illustrated in Fig. 2(a), where a single reply slot is assigned in each frame. Here the reader keeps a queue Q for storing prefixes to be used in forthcoming query requests. In each frame, the reader pops a prefix from Q and attach it to the query request. Suppose the prefix in the current frame is $q_1q_2, \cdots q_x$. If more than one tags reply (collision group of tags), the collision slot (group) is represented by a collision node n_c in the singulation tree. To arbitrate this collision, the reader pushes two 1-bit longer prefixes $q_1 \cdots q_x0$ and $q_1 \cdots q_x1$ into Q. In the forthcoming frames these new prefixes will further split this collision group into two subgroups. These two subgroups are represented by two child nodes of n_c in the tree. By repeating this process the reader can finally recognize all the tags. For example, in Fig. 2(a) only 2 idle slots but as large as 11 query requests are required to singulate 4 tags.

2) Hybrid Query Tree Scheme, HQT: As illustrated in Fig. 2(b) the HQT scheme is similar to QT one except that 4 rather than 1 reply slots are allocated in each frame. The slot selection in tags is simply implemented by referring to the next 2 bits just after the prefix bits in the tag's ID (E.g., it will reply in the first slot if these two bits are 00). After receiving replies from tags, the reader can immediately determine the next prefix for each collision subgroup based on the previous prefix and the collision slot number. For example in Fig. 2(b), the prefix for the first frame is $NULL$ and slots 0 and 3 are collision slots, so the new prefixes are determined as 00 and 11. In this example, to singulate 4 tags, only 3 requests were used, but 6 idle slots (and overall $N = 12$ slots) were generated. In this paper a generalized form of HQT scheme is considered where frame size $f = 2^k$ and tags reply based on k bits just after the prefix bits in their ID.

III. Adaptive Frame Size Singulation

In this section, we develop an Adaptive Frame Size (AFS) singulation scheme for memory-less RFID systems. The basic idea of the new scheme is to properly exploit the tag population information collected during the previous frames to



(a) QT Scheme (b) HQT Scheme (c) AFS Scheme

Fig. 2. Illustration of memory-less tag singulation schemes

Algorithm. 1 AFS Scheme

Reader Operation
*Assign a buffer Q in reader to store prefix, population info., etc.
INPUT: Maximum frame size f_{max}
1. INITIALIZATION
Set the initial prefix to $NULL$
Set the initial frame size to f_{max}
2. **for** each frame **do**
3. Read from Q the tag population info. retrieved in previous frames, and the prefix pre
4. **ESTIMATE FRAME SIZE** f based on the population info.
5. SEND REQUEST with (pre, f) to tags
6. RECEIVE REPLIES from tags
7. **if** collision(s) exist **then**
8. **INSERT NEW BUFFER ENTRIES** into Q
9. **else REMOVE STALE BUFFER ENTRIES** from Q
Tag Operation
Receive query with prefix pre and frame size f
/* x is the prefix length, $f_{bit} = log(f)$
len is the ID length
Tag ID is $t_0 \cdots t_{len}$ */
if pre match with ID
Choose the slot number with binary representation $t_{x+1} \cdots t_{x+f_{bit}}$

adaptively determine a suitable frame size for the following frames. As a result, the number of unnecessary idle and collision slots can be significantly reduced, and a good singulation performance in terms of both N and R can be achieved.

An example of using AFS algorithm is illustrated in Fig. 2(c). After receiving tags' replies to the first request, the reader singulates each collision subgroup separately. By singulating the first collision (node 2), 2 tags were recognized. To arbitrate the next collision (node 5), as the tag population of previous subgroups are 2, 0, and 0 tags respectively, so the reader chooses a small frame size (i.e. 2). With such adaptive frame size, the number of idle slots becomes 2, similar to QT, and the number of frames becomes 3, similar to HQT.

Formally, the overall AFS scheme is summarized as Algorithm 1. We can easily see that in comparison with the available schemes for memory-less RFID systems, AFS introduces some new features and modifications, like a new frame size estimation module, and also a modified buffer structure to store the tag population information collected in previous frames.

A. Frame Size Estimation

The basic idea of frame size estimation comes from the following observations: each reply slot actually corresponds to one subgroup of tag(s), and when the reader receives replies from tags, it will sequentially singulate only collision slots (if any) in a left-right order. Thus, the tag population information of "left-side" subgroups (both collision and non-collision ones) can guide the reader to choose a suitable frame size for a more efficient singualtion of "right-side" subgroups. More precisely, suppose the reader is going to singulate a collision subgroup s_i. Suppose further that the collision subgroups among subgroups $s_0, s_1, \cdots, s_{i-1}$ have been singulated and thus we know the tag populations $m_0, m_1, \cdots, m_{i-1}$ of all these i subgroups. Now, we can utilize these tag populations to get an estimate $\hat{m}_i$ of the tag population in subgroup s_i, which can guide us to choose a suitable frame size for the singulation of this subgroup. One straightforward approach to estimate $\hat{m}_i$ is to take it as the average of $m_0, m_1, \cdots, m_{i-1}$, as suggested in [17]. Notice that each collision subgroup has at least 2 tags, so we have

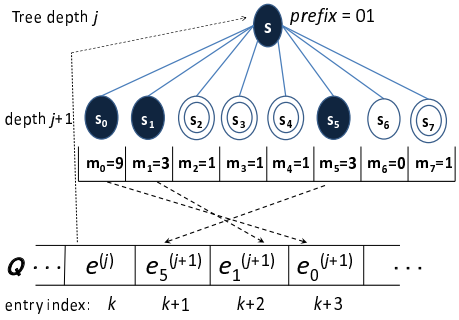
$$\hat{m}_i = Max\left\{2, \sum_{j=0}^{j-1} m_j / i\right\} \tag{1}$$

We learned from [18] that the optimal frame size should be the one that equals to the number of tags, but the current RFID systems require that frame size f should be a power of two [19], so f should be set as

$$f = Min\left\{2^{[log(\hat{m}_i)]}, f_{max}\right\} \tag{2}$$

where $[x]$ denotes the integer value nearest to x.

Consider the example in Fig. 3(a). To estimate the frame size for collisions slot s_5, the reader first gets the average population for the first five slots, which is (9+3+1+1+1)/5=3, then based on (2) it estimates the frame size as $f = 4$.



(a)

	prefix	N_c	n_s	$n_s^{(L)}$	PEI	SN
$e^{(j)}$	01	$m_0+m_1+m_5$	4	...	...	...
$e_5^{(j+1)}$	01101	...	...	3	k	5
$e_1^{(j+1)}$	01001	...	...	0	k	1
$e_0^{(j+1)}$	01000	...	...	0	k	0

(b)

Fig. 3. Illustration of the new buffer structure. (a) Correspondence between collision nodes and buffer entries. (b) Buffer entry structure.

B. Buffer Structure and Management

From the above discussion we can easily see that in addition to the prefix information required by the available QT and HQT schemes, AFS needs a new buffer structure Q to efficiently store the additional tag population information for frame size estimation (see Fig. 3). The new buffer Q consists of a queue of entries, where each entry corresponds to one collision node (i.e., a collision slot or group), which needs to be further arbitrated by a new frame. A collision group (node s) and its corresponding arbitration frame of size 8 is illustrated in Fig. 3(a). Here, node s corresponds to the kth entry in Q, while the three collision nodes (slots) in its arbitration frame correspond to the three entries after entry k. To support adaptive frame size estimation, each buffer entry now maintains the following additional elements (Fig. 3(b)).

- N_c: Overall tag population of the collision slots in the arbitration frame of a collision node.
- n_s: Number of singleton slots in the arbitration frame.
- $n_s^{(L)}$: Number of left-side singleton slots of a collision slot in its frame.
- Slot Number (SN): Slot number of a collision slot in its frame.
- Parent Entry Index (PEI): The index of a collision node's parent node entry.

The buffer entries are managed as follows.

1) Inserting New Buffer Entries: When the ongoing frame (query) results in collision(s), the last buffer entry will be updated, and for each collision slot a new entry with all information discussed above will be inserted into Q. These new entries are pushed into the buffer in a right-left order, so the collision subgroups are singulated in a left-right order. For the example in Fig. 3(a), the entry k corresponds to a collision node s in depth j of the singulation tree. Since the arbitration frame of this collision node further results in new collision slots (nodes s_5, s_1 and s_0 in depth $j+1$ of the tree), so three new entries $(k+1)$, $(k+2)$ and $(k+3)$ are inserted in Q correspondingly. The detailed process of buffer entry insertion is summarized as follows.

PROCEDURE INSERT NEW BUFFER ENTRIES

*e: The last buffer entry with index k.
*f, pre: The frame size and prefix of the current frame.
For entry e, set n_s = Number of singleton slots in the frame
for each collision slot s_i **do**
  Set $pre_{(new)}$ = CALCULATE NEW PREFIX(pre, f, i)
  Set $n_c = 0$, $n_s = -1$
  Set $n_s^{(L)}$ = number of singleton slots in the left-side of s_i
  Set $PEI = k$ /*i.e. regard e as the parent entry*/
  Set $SN = i$
  Create a new entry $e_{(new)} = (pre_{(new)}, n_c, n_s, n_s^{(L)}, PEI, SN)$
  Push $e_{(new)}$ into Q
end for

2) Removing Stale Buffer Entries: In the case the ongoing frame (i.e., arbitration frame of the last entry in Q) does not result in further collisions, the collision group corresponding to this entry has been successfully singulated and this entry is removed. Then, the next entry (i.e., the current last entry) is checked to see if its corresponding collision group has been successfully singulated. This process continues until an entry can be found whose corresponding collision group has not been singulated yet. The pseudo-code of this entry removal process is summarized as follows.

REMOVE STALE BUFFER ENTRIES

*The ongoing frame does not result in collisions.
Read out the current last buffer entry $e = (pre, N_c, n_s, PEI, \cdots)$
if $pre = NULL$ Quit Singulation
otherwise{
  Update element N_c in e's parent entry,
  $Q[PEI].N_c \leftarrow Q[PEI].N_c + e.N_c + e.n_s$
  Remove the buffer entry e}
For the new last buffer entry e_{new}
if $e_{new}.n_s \neq -1$ repeat the above process for e_{new}
otherwise exit

As we can see from the above pseudo-code, before removing an entry, the population of its corresponding collision group (i.e. $N_c + n_s$) is added to the element N_c of its parent entry. Therefore, to singulate tags in a collision slot, the overall tag population of its left side slots (which is required for the estimation process) can be derived from the sum of element $n_s^{(L)}$ in its corresponding entry and element N_c of its parent

entry[1]. For example, in Fig. 3 after subgroups s_0 and s_1 are successfully singulated (and the singulation of s_5 will start), their corresponding entries are removed, and for their parent entry k, element N_c is updated as $N_c = m_0 + m_1$. Consequently, just before removing an entry, its element N_c will be the overall tag population of its collision subgroups (e.g., in the Fig. 3, just before removing kth entry, its element N_c will be $m_0 + m_1 + m_5$).

IV. Performance Analysis

By extending the analysis in [17], we propose here a recursive calculation framework for the theoretical analysis of the new AFS scheme. With such a framework, we can also conduct the theoretical performance analysis of available QT and HQT schemes. The three basic performance parameters considered here are the expected number of slots $N(m, f)$, the expected number of requests (i.e., frames) $R(m, f)$, and the expected number of idle (empty) slots $N_e(m, f)$ we need to singulate a group of m tags with an initial frame size f.

Based on the assumption that the tags IDs are uniformly distributed in the overall tag ID space, we can establish the following Theorem regarding the evaluation of $N(m, f)$.

Theorem 1. Given an initial frame size f and a maximum frame size f_{max}, the expected number of slots $N(m, f)$, which is required by the AFS scheme to singulate $m > 1$ tags, satisfies the following recursive formula

$$N(m,f) = f + \sum_{\substack{0\le m_0,\cdots,m_{f-1}\le m \\ m_0+\cdots+m_{f-1}=m}} \left(\frac{m!}{m_0!\cdots m_{f-1}!}\right) \cdot \left(\frac{1}{f}\right)^m \sum_{i=0}^{f-1} N(m_i, f_i) \quad (3)$$

where $N(0,\cdot) = N(1,\cdot) = 0$

$$f_i = \begin{cases} 2 & \text{if } i = 0 \\ Min\left\{2^{[log(\hat{m}_i)]}, f_{max}\right\} & \text{if } i \ge 1 \end{cases} \quad (4)$$

and $\hat{m}_i$ can be derived according to (1).

Proof: The proof is removed here due to space limit. □

The Theorem 1 indicates that we can evaluate the $N(m, f)$ by recursively applying formula (3). Actually, such recursive formula can also be adopted for the evaluation of $R(m, f)$ and $N_e(m, f)$, as summarized in the following Lemma.

Lemma 1. Given an initial frame size f and a maximum frame size f_{max}, the expected number of requests $R(m, f)$ and idle slots $N_e(m, f)$, which the AFS scheme needs to singulate $m > 1$ tags, satisfy the following recursive formulas

$$R(m,f) = 1 + \sum_{\substack{0\le m_0,\cdots,m_{f-1}\le m \\ m_0+\cdots+m_{f-1}=m}} \left(\frac{m!}{m_0!\cdots m_{f-1}!}\right) \cdot \left(\frac{1}{f}\right)^m \sum_{i=0}^{f-1} R(m_i, f_i) \quad (5)$$

[1]The population information is summarized in these few elements. Thus, the extra computation (also memory) required for frame size estimation in a request is of O(1), which is negligible against the communication delay.

$$N_e(m,f) = \sum_{\substack{0\le m_0,\cdots,m_{f-1}\le m \\ m_0+\cdots+m_{f-1}=m}} \left(\frac{m!}{m_0!\cdots m_{f-1}!}\right) \cdot \left(\frac{1}{f}\right)^m \sum_{i=0}^{f-1} N_e(m_i, f_i) \quad (6)$$

where $R(0,\cdot) = R(1,\cdot) = 0$, $N_e(0,\cdot) = 1, N_e(1,\cdot) = 0$, and f_i is determined by (4) and (1).

Notice that the available HQT and QT schemes can be regarded as two special cases of the new AFS scheme, so the formulas (3), (5) and (6) can also be applied to evaluate the performances of them. For the performance analysis of the HQT scheme, we just need to set f_i as f in these formulas. For the QT scheme, since its expected number of requests is the same as its expected number of slots, so its performance can be evaluated based only on the formulas (3) and (6), where f_i and f should be set as 2.

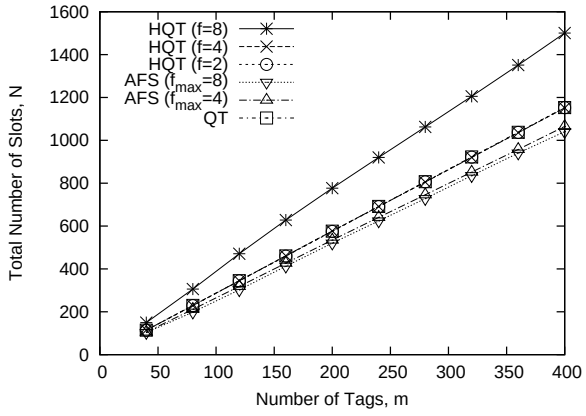
V. Numerical Results

An extensive experimental study has been performed to validate the theoretical models and also to demonstrate the performance of the new AFS scheme in terms of number of slots N, number of frames R, and number of idle slots N_e. For comparison, the results for the QT and HQT schemes are shown as well. To verify the models for AFS, simulation results are also generated. In the simulation setup m tags with mutually distinctive random IDs are located in the reader range, where the tag ID length is supposed to be 50 bits long. Given a maximum frame size f_{max} the reader probes the tags with an initial frame size set to $f = f_{max}$. Each result represents the average of 3000 experiments. The comparison results for $f_{max} = 4$ and 8, and $m = 20$ and 400 are summarized in Table. I. It is interesting to note that for all metrics the theoretical results match nicely with the simulation results, so the proposed models can be used to investigate the performance of the proposed scheme.

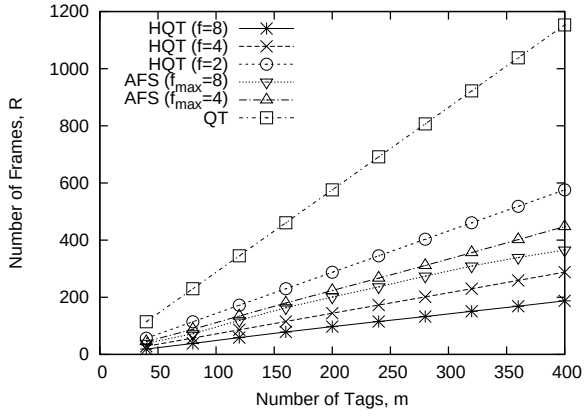
TABLE I
Model Verification

	$m = 20$		$m = 400$	
	$f_{max} = 4$	$f_{max} = 8$	$f_{max} = 4$	$f_{max} = 8$
N (Theory)	51.11	50.44	1064.61	1040.54
N (Simulation)	51.21	50.48	1064.09	1040.15
R (Theory)	21.11	20.85	447.40	365.38
R (Simulation)	21.17	20.86	447.07	365.16
N_e (Theory)	11.00	10.59	218.20	276.16
N_e (Simulation)	11.04	10.62	218.02	275.99

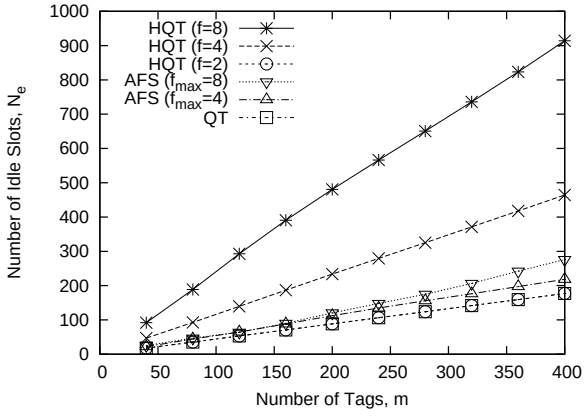
Fig. 4 depicts the results for AFS scheme (with f_{max}= 4 and 8), QT scheme, and HQT scheme (with f=2, 4, and 8) when m is varied from 40 to 400 tags. Notice that HQT with f=2 and AFS with $f_{max} = 2$ are identical schemes. From Fig. 4(a) we can observe that surprisingly the proposed scheme has smaller N than QT scheme since collisions can be avoided with expansion of the frame size. It is worth to note that in AFS as the f_{max} increases N is enhanced, while



(a)



(b)



(c)

Fig. 4. Performance as a function of number of tags, m

in HQT with increase of f, N increases dramatically. This is mainly due to the large number of idle slots caused by adopting a fixed and tag population-independent frame size in HQT (see Fig. 4(c)). From Fig. 4(b) we can see that in AFS, R is significantly lower than QT scheme, since multiple replies can be received in a frame there. This metric for AFS is close to that of HQT. However, in cases of using large frame sizes HQT shows a lower R, but which comes at the price of introducing an extensive number of idle slots.

VI. Conclusion

In this paper, we present a singulation scheme for memory-less RFID tag systems to support applications like identification or counting of tag identities. We demonstrate that by adopting an adaptive frame size, it is possible for us to achieve a both good R similar to HQT, and good E like QT such that an efficient performance of N can be achieved. We believe that the method proposed in this paper will be of great use in other applications as well, such as location discovery and tag number estimation for systems using memory-less tags.

References

[1] C. Wang, H. Wu, and N. Tzeng, "RFID-based 3-D positioning schemes," in *IEEE INFOCOM*, 2007, pp. 1235–1243.

[2] J. Zhou and J. Shi, "Localisation of stationary objects using passive RFID technology," *International Journal of Computer Integrated Manufacturing*, vol. 22, no. 7, pp. 717–726, 2009.

[3] B. Sheng, C. Tan, Q. Li, and W. Mao, "Finding popular categories for RFID tags," in *ACM MobiHoc*, 2008, pp. 159–168.

[4] L. Lu, J. Han, R. Xiao, and Y. Liu, "ACTION: Breaking the Privacy Barrier for RFID Systems," in *IEEE INFOCOM*, 2009, pp. 1953–1961.

[5] B. Knerr, M. Holzer, C. Angerer, and M. Rupp, "Slot-wise maximum likelihood estimation of the tag population size in FSA protocols," *IEEE Transactions on Communications*, vol. 58, no. 2, pp. 578–585, 2010.

[6] W. Chen, "An Accurate Tag Estimate Method for Improving the Performance of an RFID Anticollision Algorithm Based on Dynamic Frame Length ALOHA," *IEEE Transactions on Automation Science and Engineering*, vol. 6, no. 1, pp. 9–15, 2009.

[7] J. Park, M. Chung, and T. Lee, "Identification of RFID tags in framed-slotted Aloha with robust estimation and binary selection," *IEEE Communications Letters*, vol. 11, no. 5, pp. 452–454, 2007.

[8] W. Chen and G. Lin, "An efficient scheme for multiple access in a RFID system," *Proc. ICWN 2006*, pp. 160–163.

[9] M. Kodialam and T. Nandagopal, "Fast and reliable estimation schemes in RFID systems," in *ACM MobiCom*, 2006, pp. 322–333.

[10] C. Law, K. Lee, and K. Siu, "Efficient memoryless protocol for tag identification," Feb. 22 2005, uS Patent 6,859,801.

[11] ——, "Efficient Memory-less Protocol for Tag Identification," in *Proc. 4th Int'l Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, Aug. 2000, pp. 75–84.

[12] F. Zhou, C. Chen, D. Jin, C. Huang, and H. Min, "Evaluating and optimizing power consumption of anti-collision protocols for applications in RFID systems," in *Proceedings of the International Symposium on Low Power Electronics and Design*. ACM, 2004, pp. 357–362.

[13] A. Juels, R. Rivest, and M. Szydlo, "The blocker tag: Selective blocking of RFID tags for consumer privacy," in *Proc. of the 10th ACM Conference on Computer and Communications Security*, 2003, pp. 103–111.

[14] J. Myung and W. Lee, "Adaptive splitting protocols for RFID tag collision arbitration," in *ACM MobiHoc*, 2006, pp. 202–213.

[15] J. Ryu, H. Lee, Y. Seok, T. Kwon, and Y. Choi, "A hybrid query tree protocol for tag collision arbitration in RFID systems," in *IEEE International Conference on Communications*, 2007, pp. 5981–5986.

[16] P. Pupunwiwat and B. Stantic, "Unified Q-ary Tree for RFID Tag Anti-Collision Resolution," in *20th Australasian Database Conference (ADC)*, 2009, pp. 49–58.

[17] J. Kim, "A Divide-and-Conquer Technique for Throughput Enhancement of RFID Anti-collision Protocol," *IEEE Communications Letters*, vol. 12, no. 6, pp. 474–476, 2008.

[18] S.-R. Lee, S.-D. Joo, and C.-W. Lee, "An Enhanced Dynamic Framed Slotted ALOHA Algorithm for RFID Tag Identification," in *Annual International Conference on Mobile and Ubiquitous Systems*. Los Alamitos, CA, USA: IEEE Computer Society, 2005, pp. 166–174.

[19] NXP Semiconductors, "I.Code SLI-S/I.Code SLI-S SH Product Data Sheet," http://www.nxp.com/documents/data_sheet/SL113730.pdf, 2007.